# Automating File Updates via Python Algorithm

## Project Overview:

In my company, access to certain restricted content is regulated by a whitelist of IP addresses. The file "allow_list.txt" contains these authorized IP addresses. Additionally, there is a separate list of IP addresses that should be revoked, which I refer to as the remove list. To streamline the process of updating the "allow_list.txt" file and remove unwanted IP addresses, I created a Python script.

## Opening the Allow List File:

The first step of the script involves opening the "allow_list.txt" file. I started by assigning the file name as a string to a variable and then used a with statement to open the file:

In the script, the with statement is paired with the .open() function in read mode, to access the content of the allow list. This enables me to read the IP addresses stored in the file. The with keyword automatically manages resources by ensuring the file is closed after the block is exited. The function open(import_file, "r") uses two arguments: the file to be opened and the mode "r", which signifies reading. The file's contents are then stored in a variable for further manipulation.

## Reading the File's Contents:

To read the file contents, I applied the .read() method, which turns the file into a string:

When a file is opened in read mode, calling the .read() function converts its contents into a string. In my script, I assigned this string to a variable named ip_addresses. The string format allows me to process the file's contents easily in the next steps.

Transforming the String into a List:

Since I needed to remove specific IP addresses from the allow list, I converted the string into a list using the .split() method:

The .split() method divides a string into a list by whitespace by default. In this case, I converted ip_addresses into a list to facilitate IP removal.

Iterating Over the Remove List:

At the core of my script is the iteration over IPs in the remove_list. To achieve this, I employed a for loop:

The for loop repeats a section of code for each element in a sequence. The loop assigns each IP from ip_addresses to a variable called element, processing each IP one by one.

Removing the IPs on the Remove List:

The next part of the script removes any IPs in ip_addresses that are also present in remove_list. To do so, I implemented the following approach:

Within the loop, I created a condition that checks if each element exists in ip_addresses before attempting to remove it. This precaution prevents errors from trying to remove non-existent elements. The .remove() method is then applied to each IP that matches an item in the remove_list.

Updating the File with the New List:

The final step was writing the revised IP list back into the allow list file. I first had to convert the updated list back to a string, which I accomplished using the .join() method:

The .join() method consolidates list elements into a string, separated by a specific character. Here, I joined the elements using "\n" to ensure each IP appeared on a new line.

Next, I opened the file in write mode and saved the updated list:

This time, I used "w" mode in the .open() function, which indicates that I intend to overwrite the existing file content. Using the .write() method, I replaced the old IP list with the newly updated one. The file now reflects the removal of any IP addresses listed in remove_list.

Conclusion:

I developed a Python script that automates the removal of certain IP addresses from the "allow_list.txt" file. The script reads the file, converts it into a manipulable format, processes the IPs to be removed, and updates the file with the revised list. This approach enhances the efficiency of managing access to restricted content by streamlining IP removal.